SDST CM and I&T

Paula Brown

Overview

- Configuration Management:
  - Turnover Process
  - Build Process
  - Updates to CM Controlled Software
  - Supporting V1 S/W Fixes
- Testing @ TLCF:
  - Tools Being Developed to Support Testing
  - Test Data
  - Storage of Test Data

Overview (cont)

- Storage of Test Results
- Types of Tests to Be Performed
- Levels of Testing To Be Performed
- Validation/Review of Test Results
- Example of Function test for Process

Software Turnover Process

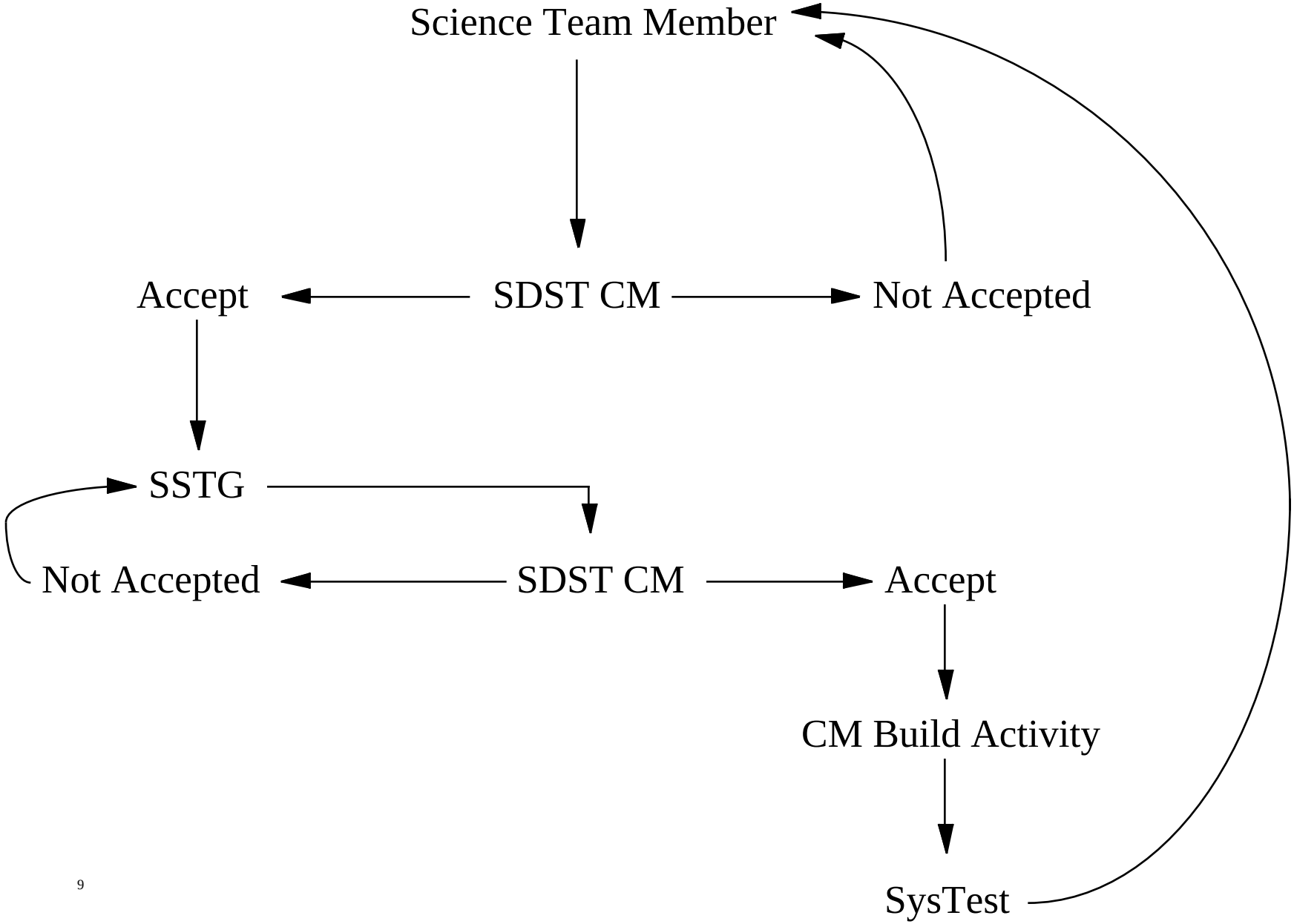
- Software turnover process has been redefined. New inboxes (mailboxes) created for turnovers.
- Inboxes have been set up to restrict unauthorized users from gaining access. Can no longer submit a turnover using the login id of ‘anonymous’.
- The following inboxes/mailboxes have been created on modisnfs3:
  - **/DEV**
    - **/IN - accessible to science team members/CM only**
    - **/CM - accessible to CM only**
    - **/OUT - accessible to STIG members/CM only**
  - **/STIGIN - accessible to STIG members/CM only**

- **Science team members:**
  - Now login using your username to modisnfs3 ftp site
  - Ftp software turnover to the corresponding directory under /DEV/IN (i.e. /MOD_PR10, MOD_PR27, etc.).
  - Upon completion execute the script ‘DEV_IN’ @ /DEV to automatically notify CM of a turnover.
    - Execution of the ‘DEV_IN' script changes the permissions of the turnover under the corresponding /DEV/IN/MOD_PR.... directory to CM. Until CM moves the files from the corresponding /DEV/IN/MOD_PR..... directory, the science team member will not be able to access this directory. The entire delivery package will be moved to the /DEV/CM/ MOD_PR.... directory.
      - To execute the script, @ /DEV enter:
        - ‘DEV_IN’ MOD_PR.... (where MOD_PR.... is the directory in which a turnover was just submitted).

- **CM Action on Science Team Turnovers:**
- CM executes the ‘DEV_CM’ script @ /DEV to move the software turnover from the /DEV/IN/MOD_PR.... directory to the corresponding /DEV/CM/MOD_PR.... directory.
- CM verifies the content of the delivery package and takes one of two actions:
  - **IF the turnover is incomplete (and discrepancies can not be resolved without a new complete turnover package):**
    - CM executes the script ‘CM_reject’ @ /DEV.
      - Execution of the script issues a notification to the science team member of turnover non-compliance and removes the turnover from the corresponding /DEV/CM/ MOD_PR.... inbox. This requires the science team member to provide a complete turnover to CM again once the problem(s) with the turnover have been resolved because both the /DEV/IN/MOD_PR.... and /DEV/CM/ MOD_PR.... inboxes will be empty.

- **If the turnover is complete:**
  - Executes the ‘CM_OUT’ script in the /DEV inbox.
    - Execution of the “CM_OUT’ script changes the permissions of the turnover under the corresponding MOD_PR.... directory to STIG, moves the software from the /DEV/CM/MOD_PR.... to the corresponding /DEV/OUT/MOD_PR.... directory, and issues a notification to STIG that software has been placed in the /DEV/OUT area for integration activities.
  - To execute the script, @ /DEV enter:
    - ‘CM_OUT’ MOD_PR.... (where MOD_PR.... is the directory in which the turnover was just verified/validated).

- **STIG Team Members:**
  - Login to modisnfs3 ftp site using individual username and change to /DEV/OUT mailbox.
  - Move the software under the corresponding /DEV/OUT/MOD_PR.... directory to the user(s) STIG work area and perform integration activities.
  - Upon completion of integration activities, login to modisnfs3 ftp site using your username and ftp software turnover to the corresponding directory in /STIGIN (i.e. /STIGIN/MOD_PR10).
  - Execute script ‘STIG_IN’ under /STIGIN to notify CM of turnover.
    - Execution of the ‘STIG_IN’ script changes the permissions of the turnover under the corresponding /STIGIN/MOD_PR.... directory to CM. Until CM moves the files from the corresponding /STIGIN/MOD_PR.... directory, the STIG team member will not be able to access this directory. The entire delivery package will be moved to the CM Build area.

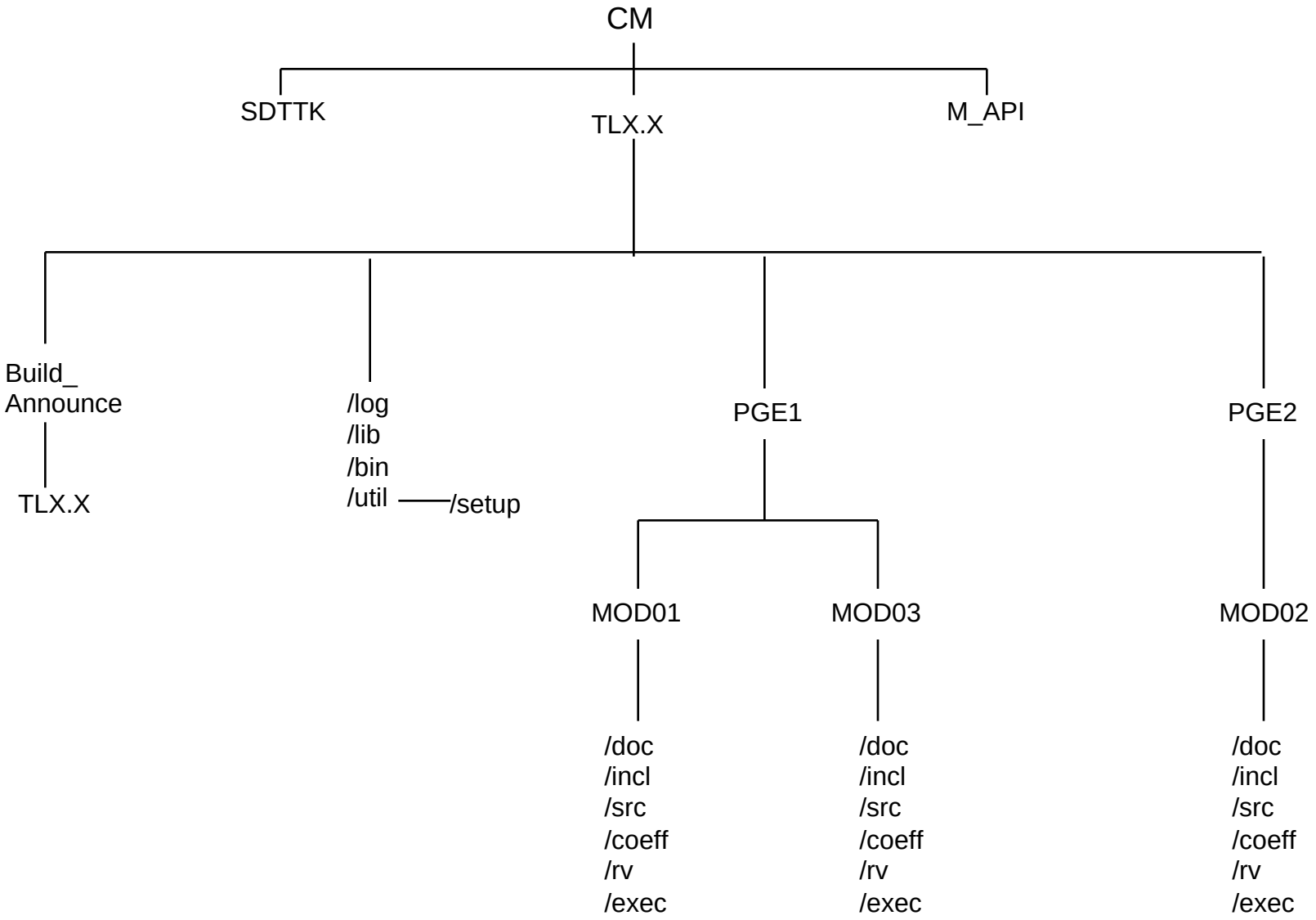

Science Team Member
Accept
SDST CM
Not Accepted
SSTG
Not Accepted
SDST CM
Accept
CM Build Activity
SysTest

CM/System I&T BUILD PROCESS

- CM will take all software received and validated by a specified date and perform a software build activity against it. This activity includes:
  - Verification of makefile compliance.
  - Verification of ability to combine the PCFs' for each process delivered.
  - Check-in of software into CM tool (ClearCase).
  - Execution of makefiles to verify ability to build executables.
  - Distribution of software into a tree structure based on a TLCF defined incremental build number. (i.e.. TL1.0, TL2.0, etc..) As each build is completed, the TLX.X number will be incremented.

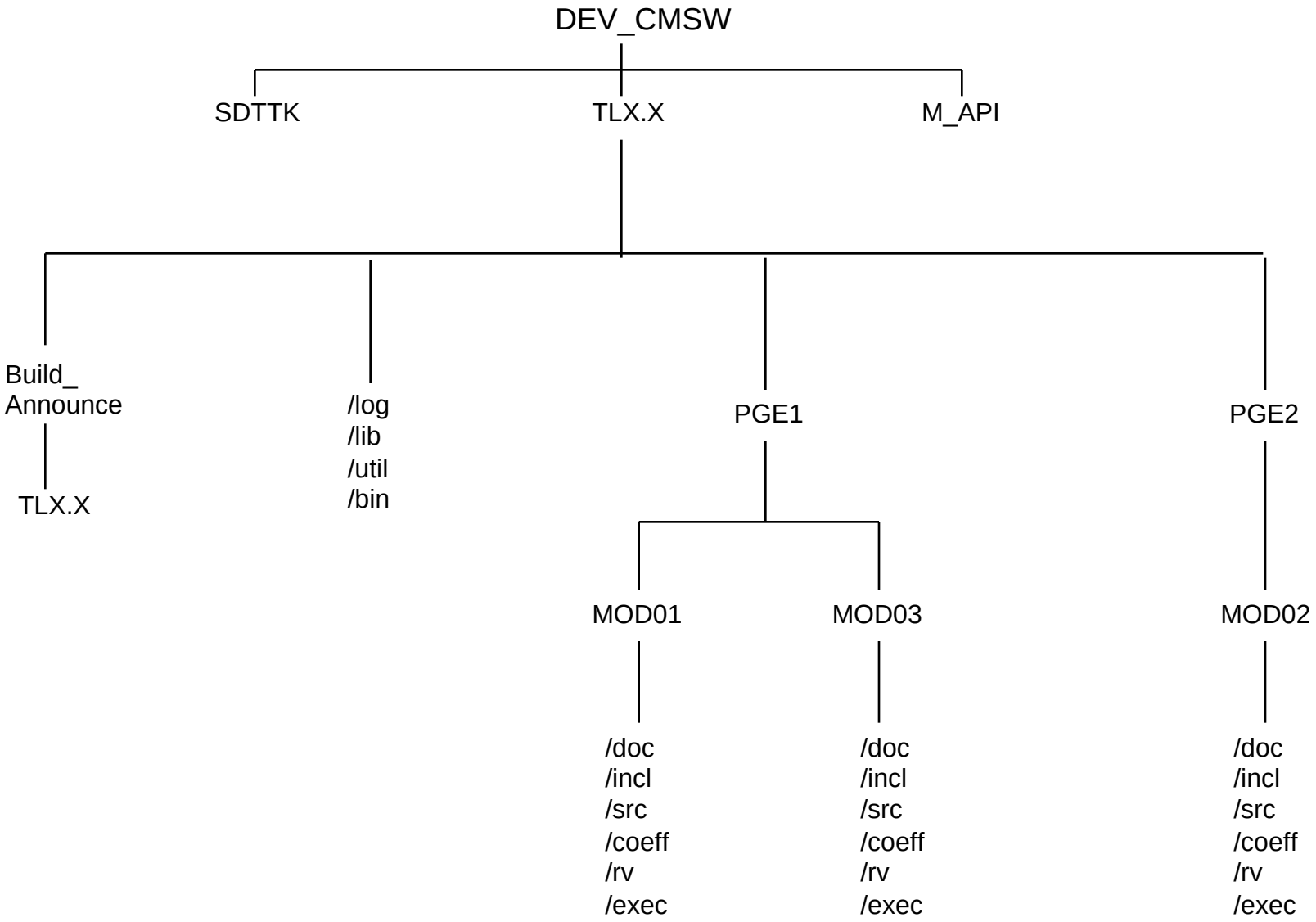
- CM Software builds will always be inclusive of all current software plus any new software received
    - Example:
        - Software build TL2.0 would include all software already existing in TL1.0 plus the additional software received and built into TL2.0 ---- TL3.0 would include all software already existing in TL2.0 plus the additional software received and built into TL3.0 ---- etc..

CM Software Tree Structure



- Upon completion of TLX.X build, CM will:
  - Publish a build announcement which lists all software included in the TLX.X build. This announcement will be posted on the software development tree under the directory ‘Build_Announce’.
  - Update the software development ftp site with the latest CM build.
    - The software development ftp site will be located on modisnfs1 @ the mount point of /DEV_CMSW. 2 gig of disk space will be allocated to support maintaining of software development area.
    - Both the science team members and STIG will have READ ONLY access to the software development area.

Software Development Tree Structure



Updates to CM Controlled Software

- When updates/modifications are required to CM controlled software, both science team members and STIG members will:
  - Have to retrieve the latest version of the CM baseline software from the software development area /DEV_CMSW and copy it to the your local work area.
  - Obtain the current UNIX checksum value(s) for the software that is to be updated from the CM baseline (/DEV_CMSW).
  - Make any appropriate updates/modifications.
    - If the updates/modifications are in response to a problem report, comments must be included in the prologue area of the corresponding software that include the problem report number and comments regarding the fix.

    - Comments are also required for enhancements that do not address a specific problem report if the software is already under CM control.
  - Obtain new checksum values for updated software being submitted to CM.
  - Provide the before and after checksum values for each updated software module in the README file when it is submitted to CM.
  - Submit updates to CM using the inboxes/mailboxes and procedures discussed in the previous slides.
- CM will:
  - Verify ‘before’ checksum values against the CM baseline to ensure that the latest version of the CM baseline file was used to perform the updates. The ‘before’ checksum verification will take place during the inbox/mailbox turnover process -- thus preventing the software from getting to the CM build process if the ‘before’ checksums do not match the CM baseline.
  - Include these updates in the next pending CM build if all inbox verifications have passed.

Supporting V1 S/W Fixes During V2 Development

- During V2 Science Software Development:
  - Testing of V1 software @ the TLCF will be ongoing.
  - Integration at the DAAC's will be ongoing.
- If a problem report is generated against the V1 software during V1 testing @ the TLCF or during any DAAC installation which is identified as a 'MUST HAVE' fix of medium or high priority, the Science Team members or STIG team members must:
  - Support the V1 activities and provide a fix/modification using the latest CM baseline version of the V1 file; documenting the problem report # in the software.
  - Determine if the V1 fix must be retrofitted into the V2 software under development and if necessary:
    - Retrofit the fix into the V2 code under development, documenting the Problem report # in the V2 software.

Why Stringent CM Control Procedures

- Without Stringent CM Guidelines:
  - System Test would have to re-run all previously passed test cases when an update to existing software was received.
  - No way to track PR and CCR resolutions from one software delivery to the next.
  - No way to ensure that previous updates included in a delivery are included in the next delivery.
  - Unable to establish any confidence level on the system software.
  - Enables system test to define a ‘baseline’ set of tet scenarios/cases to be run against each updated software build received from Configuration Management.

Testing @ The TLCF

Tools Being Developed to Support Testing

- PCF Combiner:
  - The ‘PCF Combiner’ is being developed by the SDST Test Team.
  - This tool will enable test to:
    - Combine several PCF files into a single PGE (i.e. combine MOD_PR01 and MOD_PR03 into PGE1)
    - Combine several PGE’s into a Super PGE (i.e. PGE1 (MOD_PR01 & MOD_PR03) and PGE2 (MOD_PR02) into a test PGE called PGE_L1AtoL1B.
    - Combine PCF’s together that should not process together into a single PGE. This provides a level of error checking during system test:
      - Submission of a non-valid input product to the next level of processing produces a processing error message and the process terminates gracefully.

- PCF Generator:
  - The ‘PCF Generator’ is being developed by the SDST Test Team.
  - This tool will support test automation:
    - Enables system test to create multiple executions of a single PGE or set of PGE’s in order to output multiple products or multiple days worth of products from that PGE.
      - Each PCF copy would have at least one unique character in its filename to distinguish it from another copy of that same PGE.
    - Once multiple instances of a PCF for a PGE have been made:
      - Enables system test to create a script to sequentially execute each with the appropriate PCF.

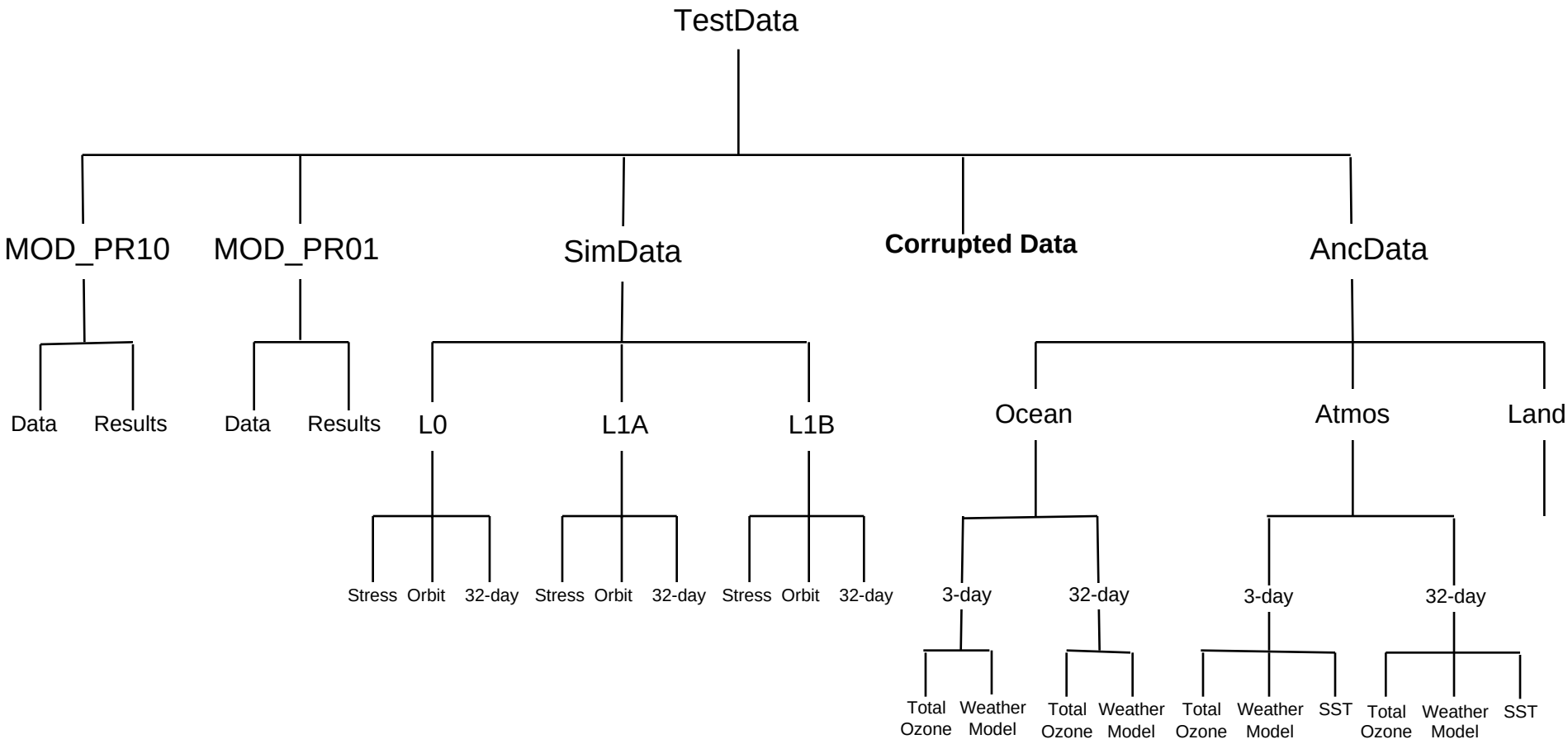
Test Data

- Types of test data:
  - Simulated Data: being generated by SDST.
  - Corrupted Data Sets: being generated by SDST.
  - Processed TestData: being provided by Science Team members and/or STIG with each MOD_PR.... submitted to CM.
  - Ancillary Data: being extracted by SDST from NASA and DAO fileservers.
  - Produced TestData: Lower level output products that are produced during SDST testing which will be used as input for higher level process testing. (i.e. product(s) produced from L1A and L1B testing will be used as inputs for L2 testing).

Storage of Test Data

- MODISNFS4:
  - Storage of Processed, Simulated, Corrupted, and Ancillary data sets will be accessible to all users (science team members, STIG, the test team, and the DAAC's) in READ ONLY mode at the mount point of /TestData on the MODISNSF4 file server.
  - In the event of shortage of disk space for test data storage, the processed data will be backed up to tape and removed from the online file server. The directory structure placeholder for that data set will remain within the tree structure with a README text file that indicates the location (physical tape/image) where the data can be found.

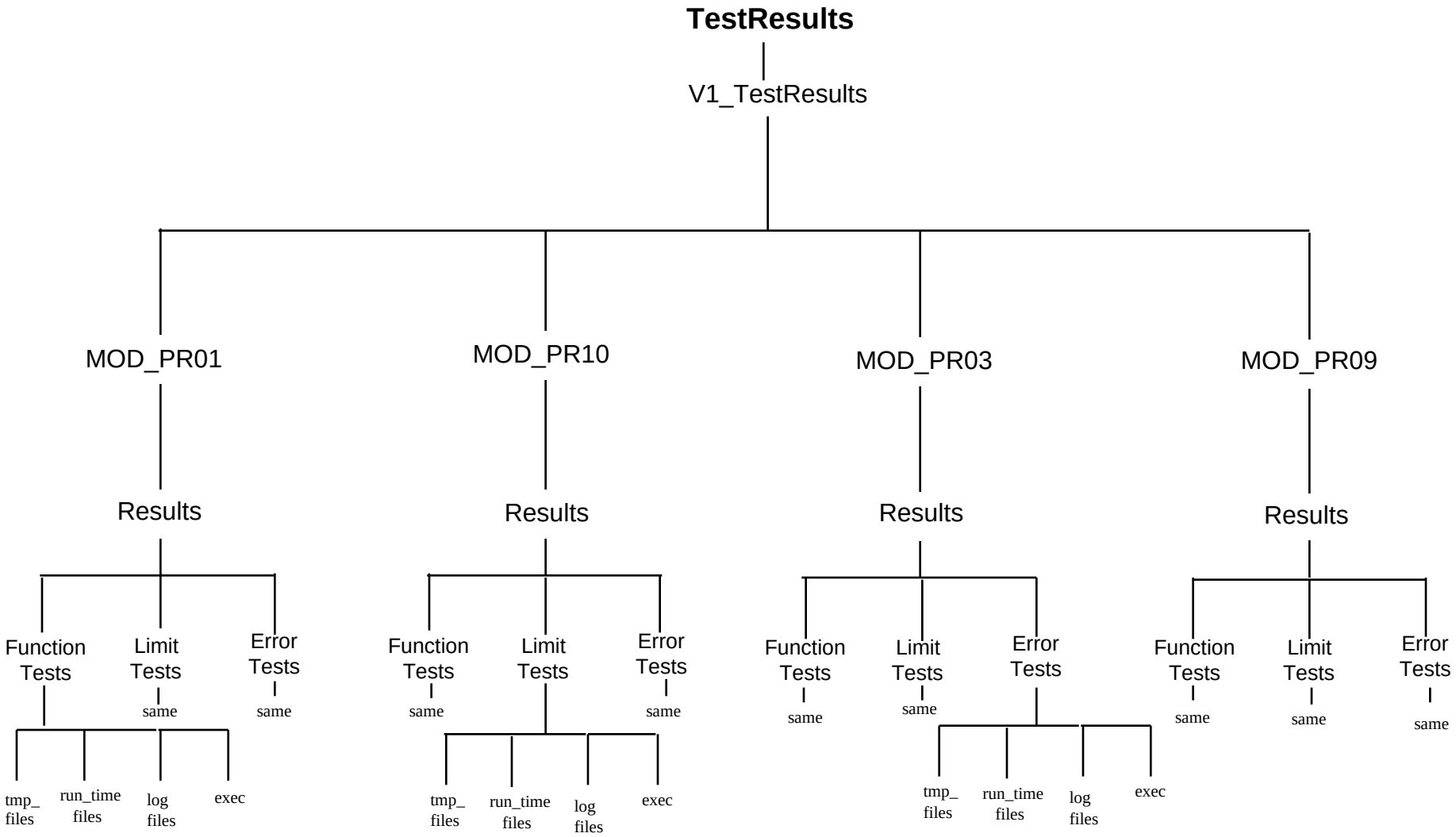
Test Data Tree Structure



Storage of Test Results (Produced Data)

- MODISNFS1:
  - Storage of Produced data sets (SDST testing results) will be accessible to all users (science team members, STIG, the test team, and the DAAC's) in READ ONLY mode at the mount point of /TestResults on the MODISNSF1 file server.
  - In the event of shortage of disk space for storage of test results, the produced data will be backed up to tape and removed from the online file server. The directory structure placeholder for that data set will remain within the tree structure with a README text file that indicates the location (physical tape/image) where the data can be found.

Test Results Tree Structure



TLCF Test Configuration

- V1 Test Configuration:
  - Performed on modispc
  - Utilizing 66gig of disk space
  - Configured with IRIX6.2 Operating System
  - Software compiled and tested in -n32 mode
  - Will utilize SDPTK version 5.1 and M-API 2.0

Types of Tests To Be Performed

- SDST will perform:
  - **Function testing**: Verifying that the software performs according to required system specifications. Internal and External interfaces are verified during function testing.
  - **Error testing**: Verifying that the software terminates gracefully or generates error messages when erroroneous conditions are injected.
  - **Performance (sizing and timing) testing**: Statistics on the amount of wall clock and CPU time each processor takes will be gathered but not verified against any system requirements.
  - **Regression testing**: Re-testing of previously tested software. Done for re-delivered (modified) software and at the Thread level.

Levels of Testing To Be Performed

- Process Testing:
  - A Function test will be performed on each individual process being delivered (i.e. MOD_PR01, MOD_PR10, etc.). Timing and sizing data (utilizing bintime only) will be gathered for each function test.
- PGE Testing:
  - PCF Combiner will be executed to combine several processes into a single PGE. Combining of processes into a single PGE may vary from the recommended set of PGE's that are to be delivered for V1 based on scheduled delivery dates for each process. Processes will be combined into the recommended PGE's for delivery to the DAAC.
  - Function and error testing will be performed and sizing/timing statistics will be gathered.

- Thread Testing:
  - Testing a combined set of processes across multiple PGE's where one PGE outputs a product (or products) that is used as the input data to the next process. (i.e. L1A to L2 to L2G, etc.)
  - PCF Generator and PCF Combiner will be used to generate and combine processes of several PGE's together (will automate granule to tile, granule to orbit, single day to multi-day tile.) where possible.
  - Function and error testing will be performed and sizing/timing statistics will be gathered if feasible. Regression testing is automatically implied at thread level testing.
- System Testing:
  - End-to-End testing within the confines of system constraints.
  - Purpose within the TLCF is to verify that proper memory allocation/ de-allocation is taking place with multiple processes running simultaneously, proper disk swapping is taking place, file sharing is not creating contention problems, runtimes are within a tolerable limit, and that valid products are produced.

Validation/Review of Test Results

- Test Results, logfiles, temporary files, and executables used during testing will automatically be posted on the modisnfs1 fileserver @ the mount point of TestResults. The SDST test team will conduct a preliminary review of the test results for test verification; however, it will be the responsibility of the science team members to review the results on the file server for scientific data validation.
- Scientists will have to examine the results from SDST testing to determine the scientific validatity of all data results produced and notifiy SDST System Test of QA review results.

Example of Function Test Script

- (1)#Run Function Test for MOD_PR10G
- (2)setenv PROGNM $1
- (3)cp /CM/TLX.X/PGE#/$PROGNM/exec/ $Prognm.pcf /systest/TLX.X/PGE#/ $PROGNM/exec/$PROGNM.tmp.pcf
- (4)setenv PGS_PC_INFO_FLE /systest/ TLX.X/PGE#/$PROGNM/exec/ $PROGNM.tmp.pcf
- (5)source /systest/TLX.X/util/setup
- (6)set_TK50_32
- (7)make -f $PROGNM.mk.clean
- (8)make -f $PROGNM.mk run

- (1)Comment which defines process under test
- (2)setting env for process under test (PR10G)
- (3)copying CM baseline files to systest env
-
-
- (4)setting toolkit env variables to the temporary PCF file
- (5)script that defines aliases for TK and MAPI
- (6)set environment for testing (TK and MAPI)
- (7)clean env for testing. Removes old logfiles and object files
- (8)run executables

- Input and Output test data pathnames will be defined within the PCF file itself.
- Input files will be read from TestData or TestResults depending on the process level under test.
- Output files will be written to TestResults/V1_TestResults tree automatically.